# Cooperative Visualization of Computational Fluid Dynamics

Michael J. Gerald-Yamasaki

RNR Technical Report RNR-92-007, March, 1992

National Aeronautics and Space Administration

**Ames Research Center**
Moffett Field, California 94035

ARC 275a (Feb 81)

# Cooperative Visualization of Computational Fluid Dynamics

Michael J. Gerald-Yamasaki

RNR Technical Report RNR-92-007, March, 1992

Supersedes RNR Technical Report RNR-91-011, March, 1991

Numerical Aerodynamic Simulation Systems Division
NASA Ames Research Center, Mail Stop T045-1
Moffett Field, California 94035-1000
yamo@nas.nasa.gov

March 10, 1992

# Cooperative Visualization of Computational Fluid Dynamics

Michael J. Gerald-Yamasaki

Numerical Aerodynamic Simulation Systems Division
NASA Ames Research Center, Mail Stop T045-1
Moffett Field, California 94035-1000
yamo@nas.nasa.gov

March 10,1992

*Abstract*

Tempus Fugit/Interview *is a computational fluid dynamics visualization application for which processing is distributed between high performance graphics workstations and supercomputers. Facilities are provided in the application for more than one user to view shared images creating a cooperative visualization environment. The way in which the computation is partitioned between the supercomputer and the workstations is critical to the capability of the application to present simultaneous, identical, animated images of fluid dynamics to more than one user.*

## 1. Introduction

Scientific visualization has become increasingly important in the analysis of large data sets. The pattern-recognition capabilities of the visual sense are utilized to analyze much greater quantities of visually transformed data than is possible with purely numeric data [7, 21].

As the visualization applications utilize the advanced features of high-performance graphics workstations and the computational power available through distributed processing with supercomputers, the individual scientist is presented with more and more information-laden images. The ability of a scientist to share the information acquired during the visual analysis process with another scientist becomes more difficult. Images created by such visualization applications are not transportable beyond the workstation without some loss of informational content.

Recently developed computer tools which facilitate collaboration have shown that a WYSIWIS ("what you see is what I see") interface is valuable. This type of interface provides for the "presentation of consistent images of shared information to all participants" [25, 26]. Such an interface to a visualization application would allow scientists to see and interact with each other's work through their workstations.

Visualization, then, becomes a communication medium and the graphics workstation a platform for the exchange of ideas.

In the study of computational fluid dynamics (CFD) visualization is used to depict the physical characteristics of computationally simulated fluid flow [16]. Visualization of unsteady fluid flow presents the problem of how to manage the computational resources required to interactively present animated images extracted and calculated over very large data sets. Cooperative visualization presents an even more daunting problem [14].

In order to provide an interactive interface for the cooperative visualization of unsteady fluid flow, the computational tasks must be partitioned between the graphics workstations and the supercomputer in a way which efficiently utilizes the strengths of each environment.

## 2. Background

In CFD the increased capabilities of supercomputers have been used to dramatically increase the size and complexity of numerical simulations of fluid flow [24]. As the size of the simulations increase, the size of the solution data also increases and can result in immense data sets representing the physical characteristics of a flow field.

Despite advances in the delivery of computational power to users of high-performance graphics workstations, there remain visualization applications for which the computational requirements can only be met by supercomputers. Distributed processing is used to provide the user with the combined capabilities of a high performance graphics workstation and a supercomputer under the control of a single application [4, 20]. The supercomputer's capabilities of great processing power, large memory, large disk storage, and fast disk access are necessary to contain, access, and calculate over, the large data sets endemic to unsteady fluid flow analysis. The high speed graphics of the workstation transform numerically calculated data into high resolution animated images of fluid flow features. The human-machine interface, also provided by the workstations, adds mechanisms for controlling the analytical tools of the visualization system.

The high resolution, three-dimensional, color, animated images which are the result of the visualization process are not transportable beyond the workstation without some loss of informational content. Much of the three-dimensional character of the images is lost without the capability to perform interactive view transformations. Recording the animated images using video is limited by the resolution of NTSC video encoding and eliminates interactive capability. Color printing or photography is comparable in color resolution and in some cases is superior in image resolution to the workstation graphics monitor. However, much of the information to be analyzed is in the animation of the images and cannot be captured with still images.

The lack of transportability of these images makes it difficult to communicate the results of the visualization process to collaborators, which in turn makes the collaboration process that much more difficult. A cooperative visualization tool, which would allow the users to simultaneously view the images as they are created, would create a shared environment for analyzing the images and facilitate the dialog so important to collaboration.

*Tempus Fugit* ("time flies") is a tool for visualizing unsteady fluid flow [12]. Processing in Tempus Fugit is distributed between a high performance graphics workstation and a supercomputer. The companion program, *Interview*, provides facilities to share the supercomputer computational environment with a second workstation creating a cooperative visualization environment.

The development of Tempus Fugit/Interview is necessarily a interdisciplinary endeavor involving CFD, graphics, distributed processing, supercomputing, and computer-supported cooperative work (CSCW). This paper focuses on the CSCW and distributed processing aspects of Tempus Fugit/Interview, discussing the other aspects as necessary to understand the overall application.

## 3. Facilitating Collaboration

One model of a collaborative environment is represented in the simple diagram in figure 1 [23]. While conversation is serial and ephemeral in nature, shared space is substantial and provides another medium for communication.


**Figure 1: Collaborative Environment Model.**

Shared access to information makes symbolic representation more concrete. Stefik, *et al.* use the chalkboard as a metaphor for a shared space for information storage [26]. The idea of WYSIWIS follows somewhat from this basic model.

With typical visualization applications a single user is "alone" with the data and the images which are the result of the visualization process. When an interesting image is produced on the graphics monitor, it's common in our laboratory to call co-workers to the monitor to see what has been produced. With the shared view of the monitor, the collaborative environment outlined above is created (figure 2). Collaborators who are in another building or another city however, cannot participate in this interaction.

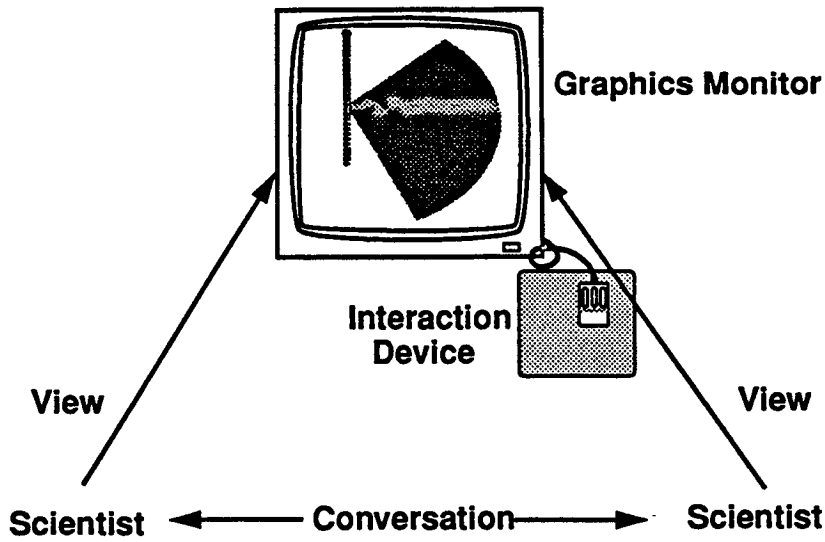


**Figure 2: "Gather Around" Collaborative Environment.**

Tempus Fugit/Interview builds on the basic model to provide an environment where distance is not a deterrent to creating the collaborative environment. For Tempus Fugit/Interview, the shared space resides on the supercomputer and the images are rendered on separate graphics workstations (figure 3).

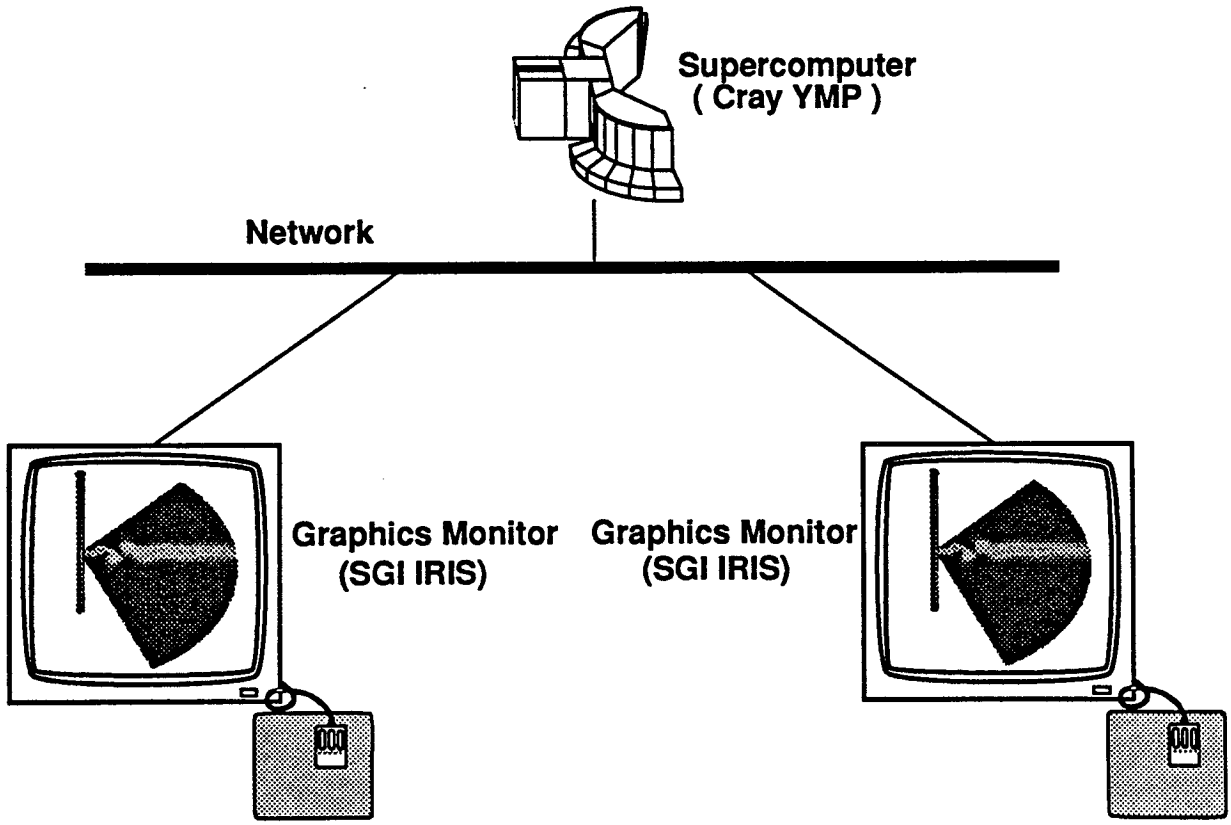


**Figure 3: Tempus Fugit/Interview Collaborative Environment.**

## 4. Previous Work

The defining characteristic of a CSCW application is, perhaps, how the user perceives the application as a tool for the sharing of information [13]. A further distinction has been made between collaboration-transparent applications, in which users share single user applications, and collaboration-aware applications, in which special features that facilitate interactions between multiple users of the application are supported [17].

A strict WYSIWIS interface in which identical images are displayed to the users might be considered a feature of a collaboration-transparent application. A looser coupling of the WYSIWIS interface allows for more flexibility in meeting the needs of a broader class of applications [8, 17, 25].

The benefits and disadvantages of various levels of coupling and collaboration awareness in applications have been studied for shared window systems [6, 17, 19]. As the technology for window systems develops, it is possible that in the future window systems will have the capability of producing the three-dimensional graphics images which are standard for CFD visualization. As it is, the graphics system and the window system are separate elements within the high performance graphics workstation architecture as exemplified by the Silicon Graphics IRIS workstation line. Three-dimensional graphics images are not transportable in the same way that windows are transportable with a virtual terminal system such as the X Window System [18]. For this reason an architecture which utilizes a virtual terminal as the primary user interface such as that used by Rendezvous [19] is not compatible with the three-dimensional graphics images used in CFD visualization.

Tempus Fugit/Interview uses a WYSIWIS interface in a flexible way with the user choosing to see precisely what the other is seeing or not as they desire. View transformations are used to point out features of complex images facilitating the sharing of the analysis of the visualization.

The partitioning of the processing among the constituent machines is an important design consideration of CSCW applications and will be discussed in detail in later sections.

## 5. The Computational Environment

One of the main objectives of the Numerical Aerodynamic Simulation (NAS) Program at NASA Ames Research Center is the provision of a comprehensive computing environment to facilitate computational aerodynamics and fluid dynamics research [1]. To this end the NAS Processing System Network (NPSN) was developed. The NPSN contains a wide range of computer systems, including two high-speed processors (currently, a Cray 2 4/256 and a Cray YMP 8/256) and a small army of Silicon Graphics IRIS workstations. Several networks provide

connectivity and a basis for network development and research. These networks include Ethernet, UltraNet, and FDDI.

The main vehicle for distributing computation between supercomputers and workstations in Tempus Fugit/Interview is Distributed Library (dlib) [29]. Like many systems which provide for distributed processing, dlib is based on the remote procedure call (RPC) model [2, 9, 27, 28]. However, unlike most of these systems, dlib was developed to provide a service which allows for a conversation of arbitrary length within a single context between client and server. The dlib server process is designed to be capable of storing state information which persists from call to call, as well as allocating memory for data storage and manipulation. While RPC protocols are frequently likened to local procedure calls without side effects, dlib more closely resembles the extension of the process environment to include the server process.

The use of dlib is much like developing a library of routines, say, an I/O library, on a local system. Application code is linked to routines in an I/O library to give access to the I/O devices controlled by the operating system, as illustrated in figure 4.

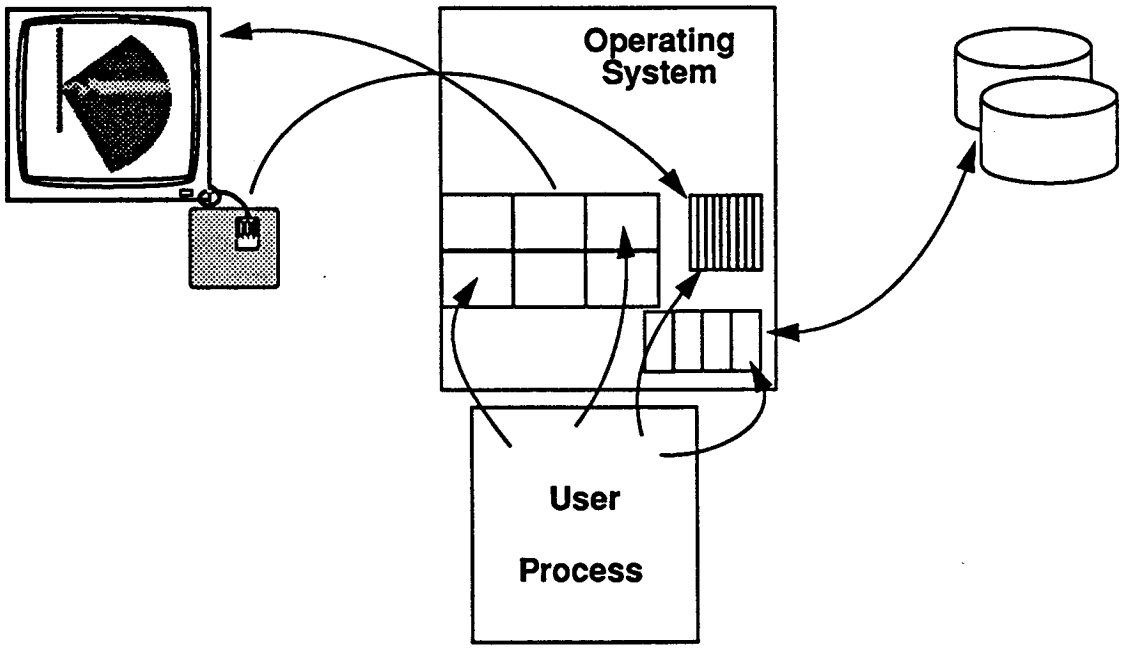


**Figure 4: I/O Library.**

Dlib provides utilities for automatic stub generation to handle the transfer of the information necessary to execute the routine in the remote environment. Due to the persistent nature of the remote environment, dlib is able to coordinate allocation and use of remote memory segments and provide access to remote system utilities. The application, through dlib, can "link" to the remote system's I/O library, for example, to utilize the remote system's I/O devices as depicted in figure 5.

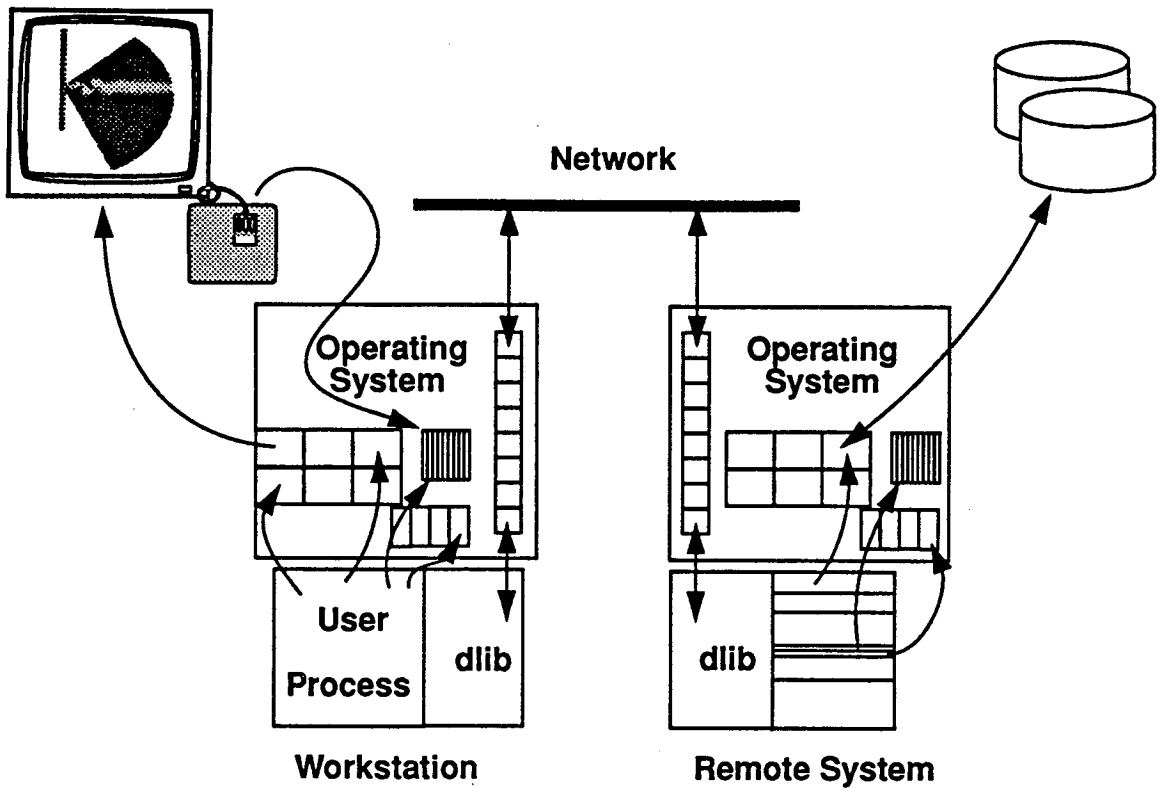


**Figure 5: Access to remote I/O devices using dlib.**

## 6. The CFD Application

The process involved in CFD research can be divided into three steps: grid generation, numerical simulation of fluid flow, and post-processing of the resulting flow solution data. A numerical grid is created describing an object and the fluid space surrounding the object. Flow solvers calculate physical properties of the flow at the nodes of the grid. The flow solution can be steady state, in which the physical properties at each node do not change over time, or unsteady, in which changes in the physical properties are observed over time.

Typical post-processing data sets consist of a grid file, containing the x-, y-, and z-coordinate values for corresponding grid nodes, and a solution file, containing the values for density, energy, and momentum for each grid node. Density and energy are scalar values while momentum is a three-dimensional vector. A steady state solution data set contains a grid file and a solution file. An unsteady solution data set contains a grid file and a solution file per time step. Typical grid sizes can be as large as several million nodes. Due to storage considerations, unsteady solution data sets are usually truncated in some manner but can still consume multiple gigabytes of storage space.

With the basic solution values of density, energy, and momentum, other physical characteristics of the flow can be calculated. Other scalar values that may be calculated include temperature, pressure, velocity magnitude, and kinetic energy. Other vector fields include velocity, vorticity, and gradients of scalar fields such as the pressure gradient. A variety of visualization techniques can be applied to these scalar and vector fields.

To get an idea of how the visualization process works in a distributed and cooperative environment, the data of the numerical simulation of the unsteady fluid flow past a tapered cylinder [15] will be used as an example. The grid for the tapered cylinder is relatively small at 128 K nodes. A data set representing 256 time steps is used for the visualization (figure 6). IRIS format and Cray format differ in word size.

Tapered Cylinder Data:

Grid: 32 x 64 x 64 nodes, 131,072 nodes, or 393,216 words

IRIS format: 1,572,864 bytes

Cray format: 3,145,728 bytes

Solution: 256 time steps for 5 physical values, 167,772,160 words

IRIS format: 671,088,640 bytes

Cray format: 1,342,177,280 bytes

**Figure 6: Tapered Cylinder Data Size.**

There are a variety of visualization techniques which can be applied to this data. One method for visualizing a data set like that of the tapered cylinder is to look at a physical characteristic of the flow over time. Calculated values for, say, velocity magnitude can be mapped to color values and displayed for a grid surface. A grid surface is a subset of the curvilinear grid (i-, j-, k-coordinates) for which one of the three components is a constant (figure 7). The area between the grid points can be rendered with interpolated color values using the Gouraud shading capability of the IRIS Graphics Library. Successive time steps can be rendered to create an animation.

The basic steps involved in creating a grid surface animation are:

Step 1. Extract data for grid surface from solution data set.

Step 2. Calculate scalar values for each node in the grid surface data.

Step 3. Map scalar values to color values.

Step 4. Render and display surface with Gouraud shaded polygons.

Step 5. Animate by displaying surfaces sequentially by time step.

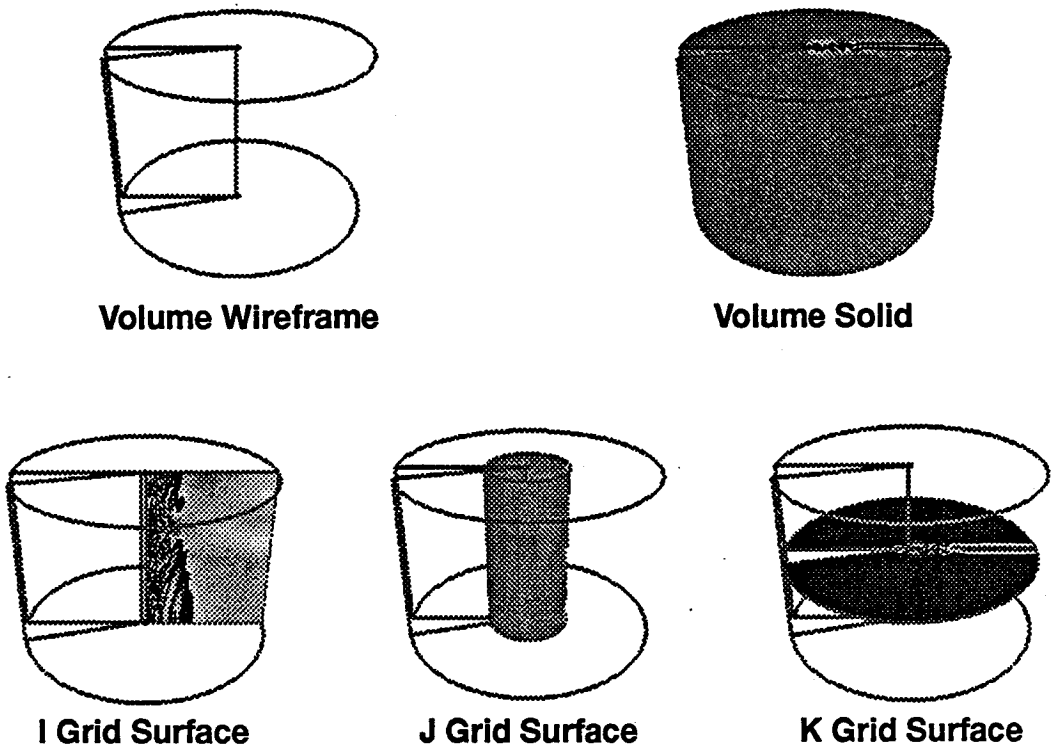


**Figure 7: Tapered Cylinder Grid Surfaces.**

## 7. Partitioning

The visualization process can be understood as the transformation of data (figure 8). For instance, the first three steps in creating a grid surface animation transforms raw data into processed data. The raw data is processed and formatted to form data which can be further transformed into images by graphics processing.

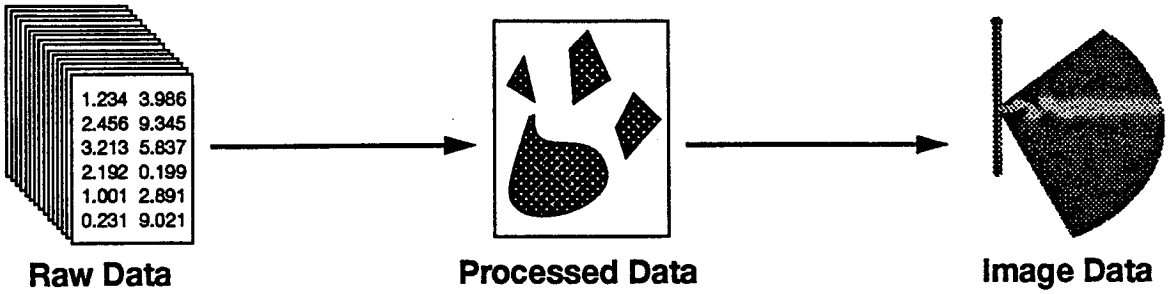


**Figure 8: Visualization Process**

Distributed processing can be applied to the visualization process by partitioning the computational tasks between a supercomputer and high performance graphics

workstations. How this partitioning is accomplished will determine the ability of the visualization system to provide a cooperative environment which best utilizes the capabilities of the graphics workstations, the supercomputer, and the network. This section describes several possible partitions: the distributed file system partition, the frame buffer partition, the distributed graphics library partition, and the geometry partition.

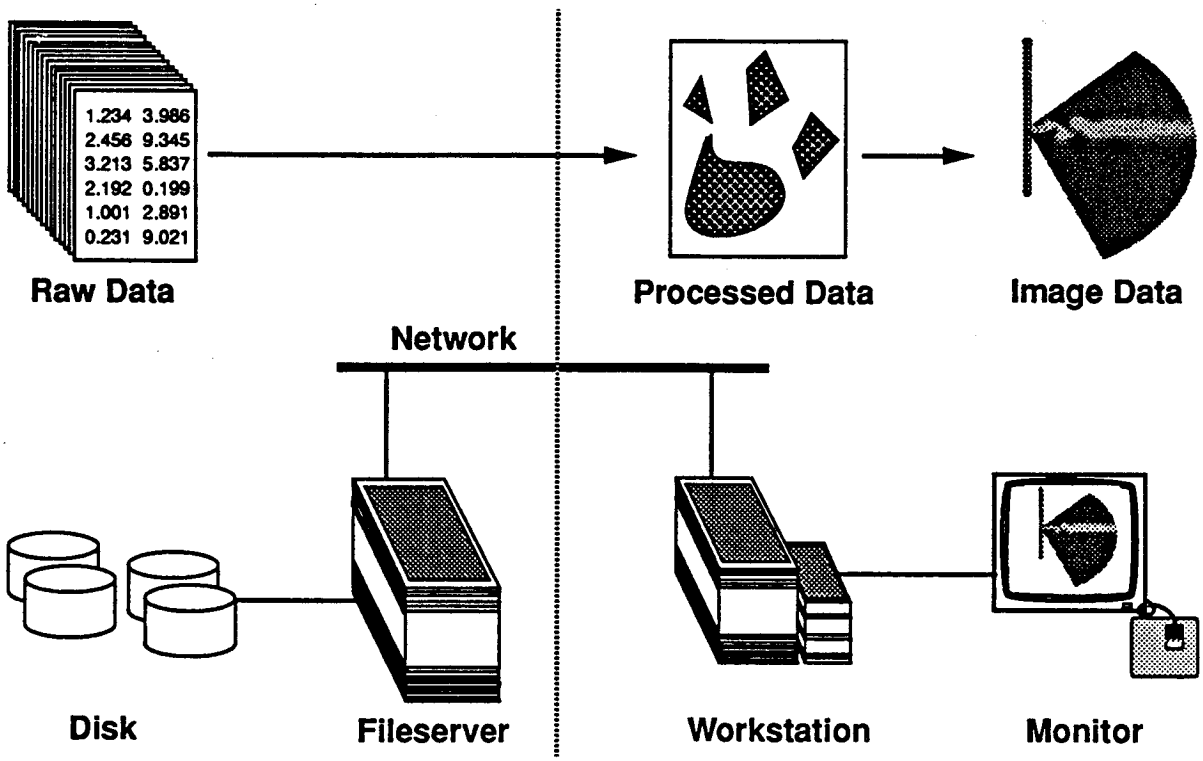


**Figure 9: Distributed File System Partition.**

Figure 9 illustrates a distributed processing partition in which the raw data resides on the server machine and is made available for processing on the client. The use of distributed file systems such as Network File System (NFS) [22] is often given as an example of distributed processing. Using distributed processing in this manner allows the client to utilize the disk resources of the server. Another example of this type of partition is when a server process creates the raw data and delivers it directly to a client process. In CFD visualization this method has been used to preview data as it is produced by a flow solver. Intermediate solutions are produced by a flow solver and transferred over the network to a visualization application instead of being written to disk.

A small advantage in saving disk access time and space is gained from this method. However, for a visualization application this partition of processing would only utilize the high disk-to-memory speed of the supercomputer, leaving the computationally intensive processing of the data to the workstation. For the cooperative element, it would be necessary to transfer copies of the large amount of data extracted from disk over the network.

Another common distributed processing partition is illustrated in figure 10. All of the processing is accomplished on one machine and the image is transferred to another for display. The image can be transferred as pixel image data or further transformed into video [11,14].

This has been found to be useful in that the image display takes place at a different machine and location than the calculation and rendering of the image, diminishing a requirement of geographical proximity of the user to the machine which is carrying out the calculation.

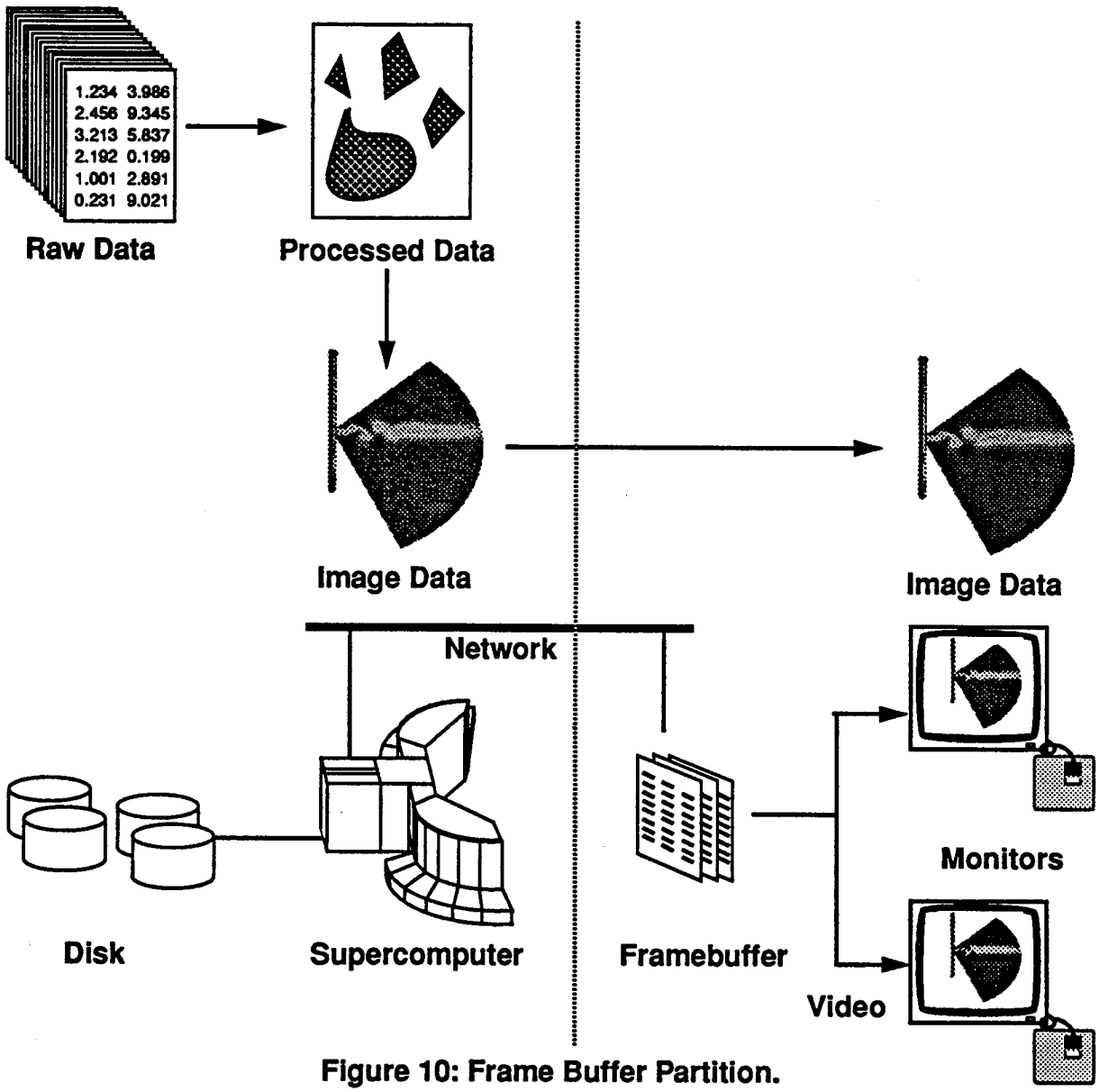


**Figure 10: Frame Buffer Partition.**

Transfer of images can turn out to be quite a large amount of data to transfer over the network, perhaps 24 bits per pixel by 1024 by 1280 pixels per frame. The data transfer rate, at 24 frames per second, would then be approximately 90 MBytes per

second. Maintaining this transfer rate to two or more workstations or frame buffers in a cooperative effort would be difficult. Using video transmission can decrease the volume of data to be transferred but only at the cost of greatly reducing the quality of the images.

While the advent of high speed networks in the gigabit-per-second range [3, 5, 10] removes a potential bottleneck in the performance of these distributed applications, high speed networks are not a panacea. Even though some networks may be capable of transferring data at a rate of one gigabit-per-second, it will be some time before workstations are capable of transferring or receiving data at that speed. Even so, image data transfer at animation speeds would be difficult to sustain.

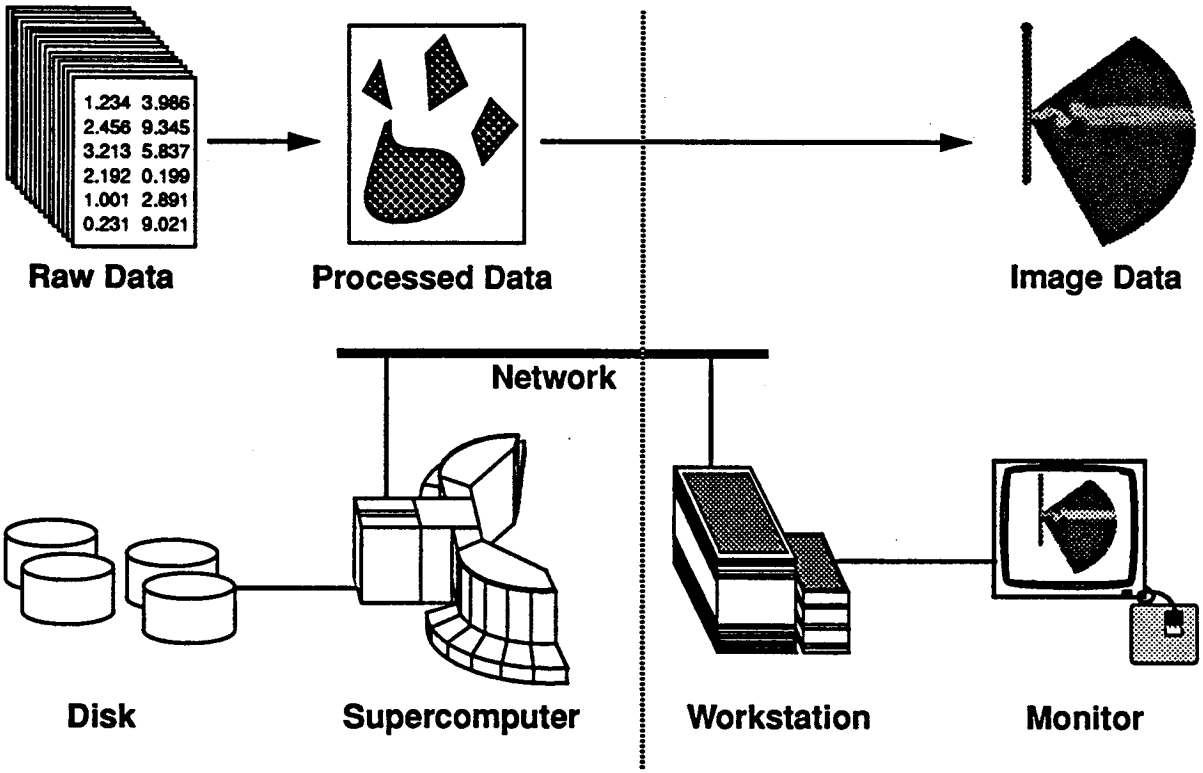


**Figure 11: Distributed Graphics Library Partition.**

A third distributed processing partition is at the transition between processed data and the creation of image data (figure 11). The data is processed and formatted to form data which can be understood by the graphics library routines. This processed data is known as geometry data. The graphics routines transform the geometry data into images. The sequence of graphics library calls may themselves be transferred over the network and executed on another machine. This partitioning has the advantage that the programming effort is carried out on the computation machine (supercomputer) and graphics calls made as though the image creation was local even though it its carried out on another machine. A special distributed graphics library is used to carry out the network transactions [24]. Network window systems are also an example of the utilization of the distributed graphics library type of

partition [18].

The distributed file system partition, the frame buffer partition, and the distributed graphics library partition are designed to have minimal impact on the basic design of an application which is being modified to distribute processing. Systems such as shared window systems have been developed which allow single machine applications to run essentially unaltered in a distributed and/or cooperative manner [17, 19].

Use of a distributed graphics library partitions the computation in a way which utilizes the strengths of the supercomputer and of the graphics workstation. It is advantageous to be able to use distributed processing which does not involve graphics. This ability is unfortunately not available with the distributed graphics library or network window system approach.

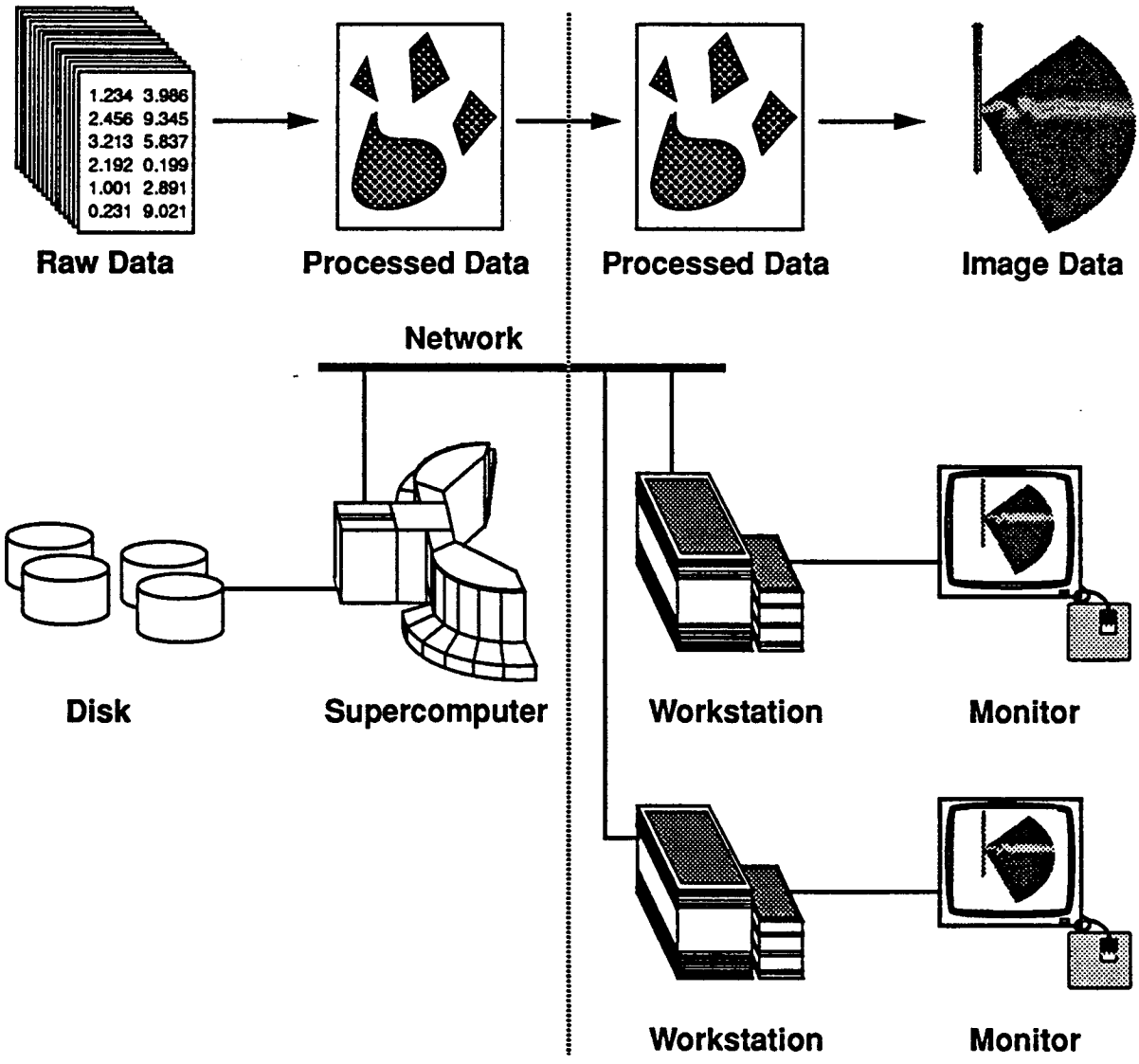


**Figure 12: Geometry Partition.**

Interactions which control the production of images from the geometry data require network transactions with a distributed graphics library partition. The network transactions and the associated processing overhead can be reduced by moving the geometry data to the workstation. The entire rendering process can then be completed on the workstation without involving the network for transferring command information or data (figure 12).

When the geometry data is transferred to two workstations, the basis for sharing the information has been established. The application of visualization techniques to the raw data creates the geometry data. With a copy of the geometry data, each workstation can render images distinct from the other workstation. By exchanging the rendering control information the two workstations can generate identical images. So, there are two levels of sharing which can be implemented with a geometry partition, geometry data and rendering controls such as view transformations.

## 8. WYSIWIS

Tempus Fugit is a stand-alone application for visualizing unsteady fluid flow. The transformation of raw data to geometry data by the application of various visualization techniques is accomplished on the supercomputer and controlled by a mouse-driven interface on the workstation. The geometry data is transferred over a connection to the dlib client on the workstation where time-sequenced animated images are produced as in the geometry partition described above (figure 12). This dlib client will be referred to as the Tempus Fugit client. The process of selecting a visualization technique to be applied, transforming raw data to geometry data, transferring the geometry data to the workstation, and rendering the data into an animated image can result, for instance, in an animation of a grid surface (figure 7). Several grid surfaces can be added to the animation (figure 13).

The dlib server executes the transformation of raw data to geometry data on the supercomputer and provides a substrate for the shared space of the collaborative environment model (figure 1).

Dlib was originally designed on a model of one client to one server. To allow multiple clients to share the server process environment, the dlib server was modified to accept more than one connection. Each connection is selected for service by the server process in the sequence that the dlib calls are received. The dlib calls are executed by the server in a single process environment as though there were only one client.

The Tempus Fugit client will have been active for an arbitrary length of time when Interview is invoked. As such, the image the Tempus Fugit client is presenting may contain a number of grid surfaces. Interview creates a connection to the dlib server on the supercomputer and to the Tempus Fugit client (figure 14). The Tempus Fugit

client maintains a list of descriptions of the grid surfaces it is currently viewing. Upon request this list is sent to the Interview client. From this list, the Interview client has the information to transfer the geometry data from the server process using dlib calls in the same way that the Tempus Fugit client received the data.

The Interview client is now able to present images created from the same geometry data as the Tempus Fugit client. Interactive controls for view transformations and animation sequencing are individual to each client. Consequently, the two clients at this point are viewing the same three-dimensional geometry data but may have different viewing perspectives.

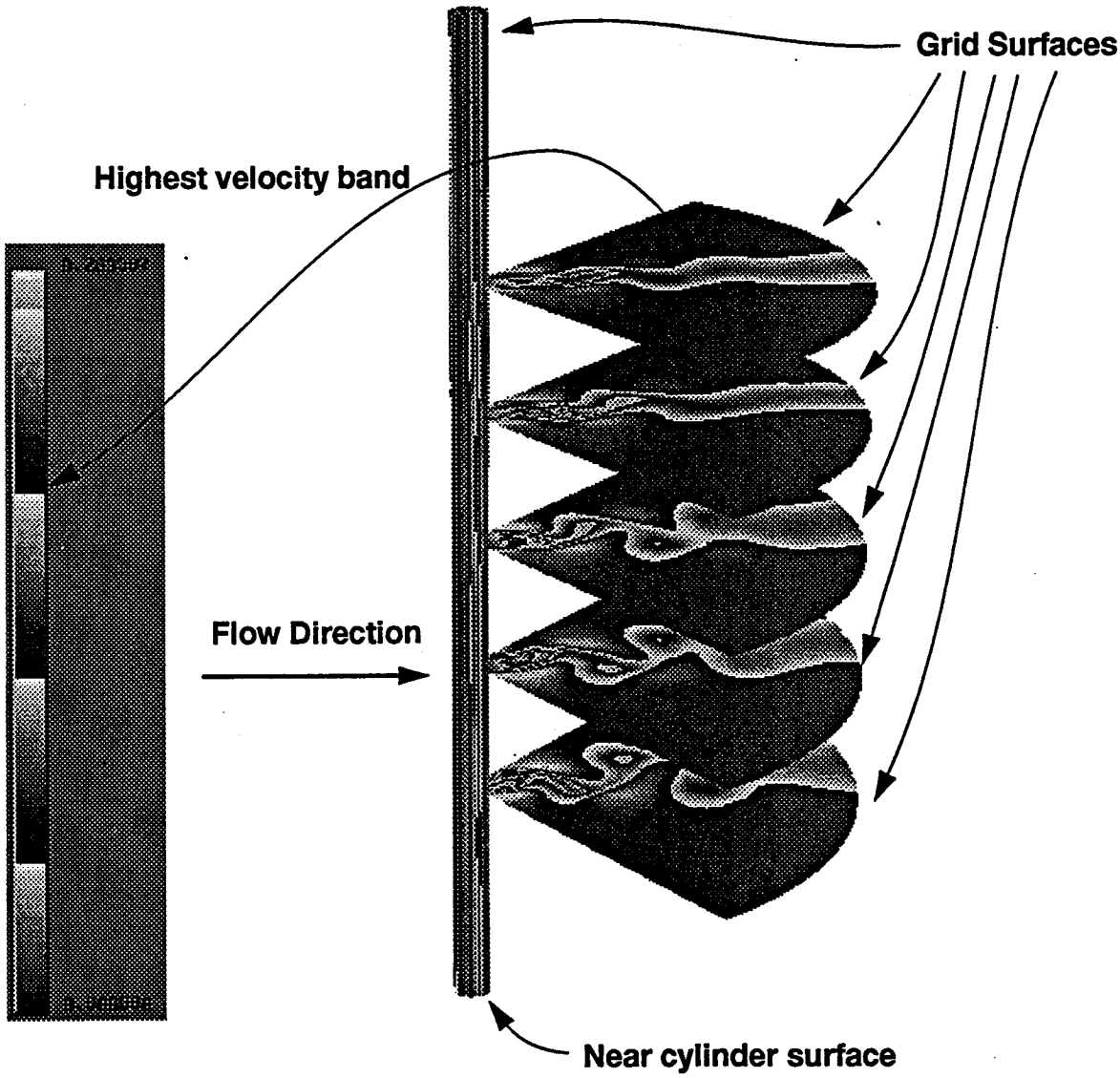


**Figure 13: Grid Surfaces Shaded by Velocity Magnitude.**

The ability to individually view the same geometry data is analogous to two people

looking at a three-dimensional object, say, an open book. One person can see the title and author on the front cover, while the other can read the pages. While their views are different, there is a shared context for conversing.

To present an identical image on both monitors simultaneously, rendering controls such as view transformations and animation sequencing must be consistent. The graphics transformation matrix controls the mapping between geometry data and the screen representation. In order for identical images to be viewed by both clients, the rendering control information of one client is sent to the other client. The receiving client loads the transformation matrix into its graphics pipeline creating an image identical with the viewing perspective of the sending client. The animation sequencing information is used to synchronize the animation frame by frame.

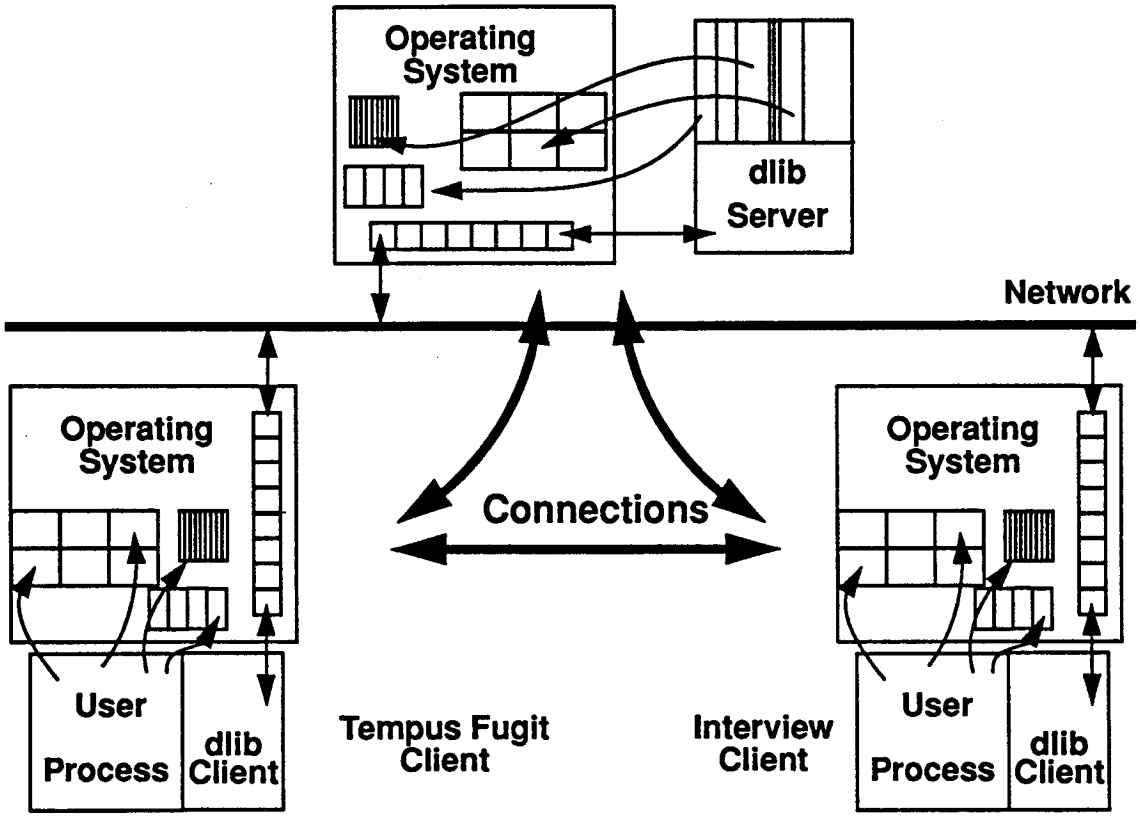


**Figure 14: Tempus Fugit / Interview Software Architecture.**

Each client has a set of mouse-driven view transformation controls for rotation, pan, and zoom. To see what is being presented by the other client "tracking mode" is selected. This results in a message sent to the other client to begin sending the rendering control information. The other client continuously updates the rendering control information until tracking mode is deselected and "detached mode" is entered. While in tracking mode, view transformation controls are disabled. While in detached mode, the client is able to control the viewing perspective.

## 9. Discussion

Shared window systems place an emphasis on sharing existing single-user applications. As such, a major design consideration is the use of a replicated or a centralized computer conference architecture [6, 17]. Tempus Fugit/Interview is an application which has a requirement for a distributed environment independent of the facility of sharing the application in a CSCW environment. Consequently, the major design consideration of the partitoning of processing is predicated on both the performance requirements of the single-user distributed application, Tempus Fugit, and the requirements of operating in cooperation with a companion program, Interview.

Tempus Fugit/Interview is a combination of the replicated and centralized computer conferencing architectures. The server portion which operates on the supercomputer can be viewed as a centralized application when it accepts inputs and distributes outputs to both workstation clients. The server "application" is to transform raw CFD data into geometry data. Once the geometry data has been transferred to the clients, the clients behave in a manner similar to a replicated computer conference architecture. The transformation of geometry data to animated images is a process which is replicated by both clients. When a client is in tracking mode, exchange only occurs on input with both clients generating output.

The difficulty associated with a replicated architecture of maintaining consistency across replicants is avoided with the hybrid architecture of Tempus Fugit/Interview. The consistency is maintained by the centralized server. The advantage of a replicated architecture in providing for activities which are unaffected by other participants in the environment is made available to the Tempus Fugit/Interview client when not in tracking mode.

The performance advantage of a replicated architecture is utilized by Tempus Fugit/ Interview when in tracking mode. The transformation matrix and other rendering control information is very compact and can be transferred between workstations well within the animation frame rate of between ten and thirty frames per second on networks of modest speed such as Ethernet. By replication of the process required to transform geometry data into images, only the rendering control information need be transferred between the workstations to provide the WYSIWIS function.

## 10. Conclusion

The way in which computation is partitioned between constituent processors is an important design consideration for distributed and cooperative applications. Many applications have been implemented for use on a single machine before it becomes desirable to operate the application in a distributed and/or cooperative environment. For an application such as the visualization of CFD, processing can be partitioned in ways which require minimal modification to a single processor implementation.

However, it is difficult for such partitions to take advantage of the full range of facilities available in a distributed environment. The advocacy for either a centralized or replicated architecture for computer conferencing applications is predicated on a desire to require minimal modification to single-user applications. This is a prudent approach for applying CSCW concepts to existing applications. The constraints implicit in making a choice between a centralized or replicated architecture do not apply, however, to applications initially developed with a distributed and/or cooperative environment in mind. For these newly developed applications, efficiencies in the passing of control, in providing shared and private contexts, and in providing computational services, should guide which elements should be a centralized resource and which should be replicated.

## 11. Acknowledgments

The author would like to thank E. Lisette Gerald-Yamasaki, Steve Bryson, and Sam Uselton for reviewing early versions of this manuscript and suggesting numerous improvements to the final presentation.

## 12. References

[1] Bailey, F. R. Status and projections of the NAS program. In *Computational Mechanics - Advances and Trends.* A. K. Noor, editor New York: American Society of Mechanical Engineers, 1986, 7-21.

[2] Birrell, A. D., and Nelson, B. J. Implementing remote procedure calls. *ACM Trans. on Comp. Sys.* 2, 1 (Jan. 1984), 39-59.

[3] Chlamtac, I. and Franta, W. R. Rationale, directions, and issues surrounding high speed networks. 78, 1 (Jan. 1990), 94-120.

[4] Choi, D. and Levit, C. Implementation of a distributed graphics system. *Internat. J. Supercomput. Appl.* (Winter 1987), 82-95.

[5] Clinger, M. Very high speed network prototype development: Measurement of effective transfer rates. In a report to NASA Ames Research Center in satisfaction of Contract #NAS2-12332/CTO#9, (Oct. 1989).

[6] Crowley, T., Milazzo, P., Baker, E., Forsdick, H. and Tomlinson, R. MMconf: an infrastructure for building shared multimedia applications. In *Proc. CSCW '90* (Los Angeles, CA, Oct. 7-10, 1990) New York, NY: ACM (Order No: 612900), 329-342,

[7] DeFanti, T. A., Brown, M. D., and McCormick, B. H. Visualization - Expanding scientific and engineering research opportunities. *Computer* 22, 8 (Aug. 1989),

12-25.

[8] Dewan, P. and Choudhary, R. Flexible user interface coupling in a collaborative system. In Proc. CHI '91 (New Orleans, LA, Apr. 27-May 2, 1991) New York, NY: ACM (Order No: 608910), 41-48.

[9] Dineen, T. H., Leach, P. J., Mishkin, N. W., Pato, J. N., and Wyatt, G. L. The network computing architecture and system: an environment for developing distributed applications. *Proceedings of Summer Usenix* (June 1987), 385-398.

[10] Farber, D. Gigabit network testbeds. *Computer* 23, 9 (Sep. 1990), 77-79.

[11] Fowler, Jr., J. D., and McGowen, M. Design and implementation of a supercomputer frame buffer system. In *Proc. Supercomputing '88* (Orlando, Florida, Nov. 14-18, 1988) Washington, DC: IEEE Computer Society Press (Order No. 882), 140-147.

[12] Gerald-Yamasaki, M. Interactive and cooperative visualization of unsteady fluid flow. *NAS Applied Research Technical Report*

[13] Grudin, J. CSCW: the convergence of two development contexts. In Proc. CHI '91 (New Orleans, LA, Apr. 27-May 2, 1991) New York, NY: ACM (Order No: 608910), 91-97.

[14] Haber, R. B., and McNabb, D. A. Eliminating distance in scientific computing: an experiment in televisualization. *Internat. J. Supercomput. Appl.* 4, 4, (Winter, 1990), 71-89.

[15] Jespersen, D. and Levit, C. Numerical simulation of flow past a tapered cylinder. American Institute of Aeronautics and Astronautics (AIAA) paper 91-0751. AIAA 29th Aerospace Sciences Meeting. (Reno, Nevada, Jan. 7-10, 1991).

[16] Lasinski, T., Buning, P., Choi, D., Rogers, S., Bancroft, G. and Merritt, F. Flow visualization of cfd using graphics workstations. *Proc. AIAA 8th Computaional Fluid Dynamics Conf.* (Honolulu, Hawaii, June 9-11, 1987) AIAA Paper 87-1180, 814-820.

[17] Lauwers, J. C. and Lantz, K. A. Collaboration awareness in support of collaboration transparency: requirements for the next generation of shared window systems. In Proc. CHI '90 (Seattle, WA, Apr. 1-5, 1990) New York, NY: ACM (Order No: 608900), 303-311.

[18] O'Reilly & Associates Inc. *X Window System Series* Sebastapol: O'Reilly, 1990.

[19] Patterson, J. F., Hill, R. D., Rohall, S. L. and Meeks, W. S. Rendezvous: an

architecture for synchronous multi-user applications. In *Proc. CSCW '90* (Los Angeles, CA, Oct. 7-10, 1990) New York, NY: ACM (Order No: 612900), 317-328,

[20] Rogers, S. E., Buning, P. G., and Merrit, F. J. Distributed interactive graphics applications in computational fluid dynamics. *Internat. J. Supercomput. Appl.* 1, 4 (Winter 1987), 96-105.

[21] Salzman, D. Visualization in scientific computing: Summary of an NSF-sponsored panel report on graphics, image processing, and workstations. *Internat. J. Supercomput. Appl.* 1, 4 (Winter 1987), 106-108.

[22] Sandberg, R. The sun network file system: design, implementation, and experience. Sun Microsystems, Inc. (1986).

[23] Schrage, M. *Shared Minds.* New York: Random House, 1990

[24] Silicon Graphics Computer Systems. Using the distributed graphics library. *4-Sight Programmer's Guide* Document Number 007-2001-030 (1990).

[25] Stefik, M., Bowbrow, D. G., Foster, G., Lanning, S. and Tatar, D. WYWIWIS revised: early experiences with multiuser interfaces. ACM Trans. on Office Info. Sys. 5, 2 (Apr 1987), 147-167.

[26] Stefik, M., Foster, G., Bobrow, D. G., Kahn, K., Lanning, S., and Suchman, L. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Commun. ACM* 30, 1 (Jan. 1987), 32-47.

[27] Sun Microsystems. *Request for Comment #1057* Network Working Group (June, 1988).

[28] Xerox Corporation. Courier: the remote procedure call protocol. *Xerox System Integration Standard (XSIS) 038112*, (Dec. 1981).

[29] Yamasaki, M. Distributed library. *NAS Applied Research Technical Report* RNR-90-008 (Apr. 1990).